République du SENEGAL

Université Cheikh Anta Diop de DAKAR

Ecole Supérieure Polytechnique

Mémoire

Master Informatique Modélisation et Simulation des Systèmes Complexes

Plateforme de Modélisation et de Simulation des Systèmes Complexes MIMOSA

Analyse et conception d'une interface avec les Systèmes d'Informations Géographiques

Par

Aboubacar CISSE

Encadré par

Pr Jean Pierre MULLER

Mr Alassane BAH

Mr Grégoire LECLERC

Année universitaire 2004/2005

Sommaire

Introduction

Les outils intégrés d'exploration, de représentation, d'explication des phénomènes spatiaux et temporels sont nécessaires à la gestion de l'environnement. Ces outils sont principalement destinés aux chercheurs, décideurs, experts… Un réseau de compétences s'est constitué autour des besoins de ces utilisateurs pour créer une nouvelle plateforme générique de Modélisation : MIMOSA.

Mimosa permet une libre définition des formalismes dans lesquels s'expriment les modèles. Pour ce faire, elle met à disposition des notions génériques, une boite à outils asémantiques, permettant de décrire les différents formalismes pertinents à chaque modélisation. La définition de formalisme constitue par conséquent une première étape dans la modélisation. Une fois les formalismes définis, le modélisateur a la possibilité de décrire les catégories d'objet dont il veut parler, il s'agit du discours sur les catégories. Dans la dernière étape le modélisateur utilise les types, les catégories qu'il s'est donné dans le niveau précédent pour construire un ou plusieurs modèles.

Nous ferons donc dans une première partie un état de l'art sur les plateformes de représentation de connaissances, ensuite, une deuxième partie permettra de décrire la plateforme Mimosa et ses différentes spécificités par rapport aux autres plateformes de modélisation et de simulation. Enfin, nous terminerons par un descriptif de l'extension vers les Systèmes d'Information Géographiques réalisée dans le cadre de cette étude-ci.

Chapitre 2

Etat de l'art sur les plateformes de représentation de connaissances

Swarm

La plateforme de représentation de connaissances la plus connue est Swarm. Elle a été réalisée par Chris Langton dans le cadre du Santa Fe Institute et s'est élargie ensuite à une communauté de développeurs. Elle se présente comme un ensemble de bibliothèques de classes écrites en Objective C. Cette plateforme est largement utilisée par les anglo-saxons grâce à son efficacité en termes de temps de réponse et donc à sa capacité à gérer des simulations de taille importante [1].

Cependant Swarm présente un certain nombre d'inconvénients en ce qui concerne l'expressivité qui est très difficile à mettre en œuvre par un non informaticien. Cela fait de Swarm une plateforme essentiellement utilisée pour des simulations de phénomènes élémentaires nécessitant une grande vitesse d'exécution et des modèles pas très complexes [2].

Enfin cette plateforme n'a pas vraiment été conçu pour modéliser des phénomènes faisant intervenir à la fois des êtres biologiques et des individus sociaux.

Madkit

MadKit (Multi-Agents Developpement Kit) est une plateforme multi-agents modulaire et scalable écrite en Java et conçue selon le modèle d'organisation Alaadin AGR (Agent/Group/Role): des agents sont situés dans les groupes et jouent des rôles. MadKit est développée en 1996 par Olivier GETKNECHT et Michel FERBER au Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) de l'Université Montpellier II. Il s'agit d'une plate-forme libre pour l'utilisation dans l'éducation. Un moteur d'exécution est utilisé dans MadKit où chaque agent est construit en partant d'un micro-noyau. Chaque agent a un rôle et peut appartenir à un groupe. MadKit est doté d'un

environnement de développement graphique qui permet facilement la construction des applications [3].

Le modèle Agent/Groupe/Rôle

L'architecture de la plateforme MadKit a été construite selon le modèle Agent/Rôle/Groupe, développé dans le contexte du projet Alaadin. MadKit implémente et utilise ce modèle pour ses différentes tâches de développement des agents. Dans ce modèle, les concepts organisationnels tels que: les groupes, les rôles, les structures, les dépendances, etc. sont considérés comme les briques de base qui permettront le développement des systèmes scalables et hétérogènes.

Agent

Le modèle n'impose aucune contrainte sur l'architecture interne des agents. Un agent est désigné comme étant une entité active communicante qui joue des rôles dans des groupes. Cette définition est générale afin de permettre aux développeurs de spécifier le modèle le plus adapté aux agents relatif à leur application (l'architecture interne par exemple).

Group

Les groupes sont les ensembles atomiques d'agrégation des agents, chaque agent fait partie d'un ou plusieurs groupes. Dans sa plus simple forme, un groupe sert seulement à étiqueter des agents. Dans une forme plus développée, et en conjonction avec les rôles, un groupe peut représenter n'importe quel système multi-agents. Les groupes peuvent être en chevauchement, car un agent peut être membre de n groupes en même temps.

Role

Un rôle est une représentation abstraite de la fonction, du service ou de l'identification d'un agent à l'intérieur d'un groupe. Chaque agent peut jouer plusieurs rôles dans le même groupe, un rôle peut être attribué à un agent à la demande de ce dernier.

La propriété de chevauchement des groupes nous permet de concevoir un monde plat ou les agents ne sont pas organisés d'une manière atomique et n'appartiennent pas à des modèles et structures rigides.

Dans chaque groupe, un agent particulier est désigné pour être le représentant de ce groupe lors des contacts avec les autres groupes, il est considéré comme le porte-parole de tous les agents membres de ce groupe.

Architecture de madkit

L'architecture de MadKit est constituée des 3 principaux composants: le micro-noyau, l'agentification des services et le modèle des composants graphique.

MadKit est constitué d'un ensemble de packages de classes JAVA qui implémentent le noyau agents, un environnement de développement graphique ainsi que des modèles standard des agents.

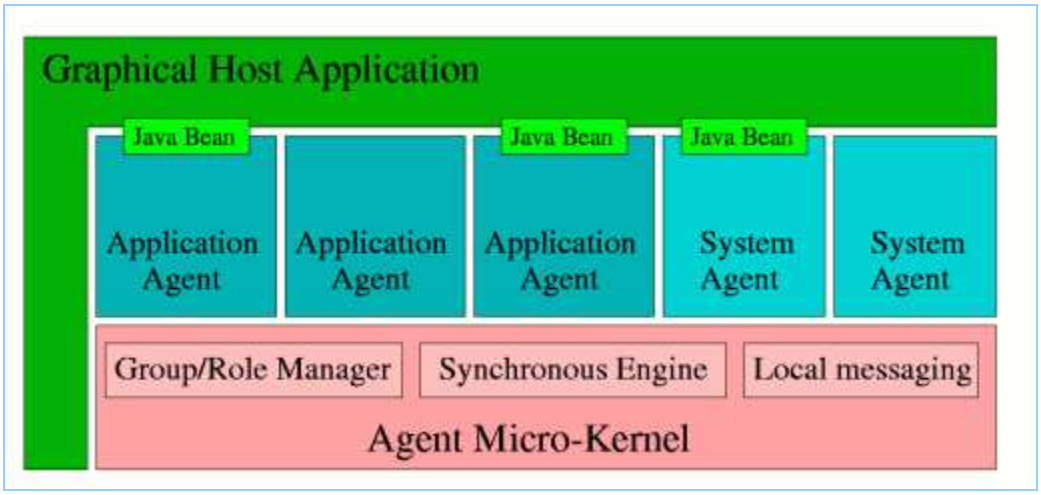

Figure 1: Architecture de madkit

La philosophie de MadKit est basée sur l'idée d'utiliser le plus possible la plateforme pour ses propres besoins de gestion. Aucun service autre que ceux offerts par le micro-noyau n'est utilisé par les agents. Les groupes des agents ont été proposés dans d'autres plateformes, mais ils manquent de la capacité de gérer des groupes multiples et des rôles différents pour le même agent au sein d'un groupe.

Le micro-noyau de MadKit est petit (moins de 40 kb) et optimisé, il offre les opérations de base qui permettent le déploiement des agents.

Le noyau de MadKit a pour charge les taches suivantes :

Le control des groupes et des rôles:

Du moment ou la plupart des possibilités d'extension et d'interopérabilité sont basées sur la couche organisationnelle, la manipulation des groupes et des rôles est assurée au niveau le plus bas du noyau afin de fournir cette fonctionnalité à n'importe quel agent. Le noyau maintient des informations sur les membres des groupes ainsi que les rôles joués dans chaque groupe. Ces informations lui permettent de faire des contrôles et vérifier la validité des demandes des agents.

La gestion de cycle de vie des agents:

Le noyau se charge aussi de la création et de la destruction des agents, et maintient des tables de référence de tous les agents lancés, il assigne une adresse globale à chaque agent, constituée de l'adresse de noyau et l'adresse de l'agent dans le noyau local. On peut noter ici que cette adresse peut être utilisée comme le standard de l'adressage des agents.

Passage des messages locaux:

Le passage des messages entres les agents locaux est effectué par le noyau, l'envoie d'un message est réalisé par la copie de ce message dans le tampon de l'agent récepteur.

Le noyau est doté d'un agent particulier, appelé Agent Noyau, chargé de gérer les opérations de noyau, cet agent est lancé au démarrage.

CORMAS

Généralités

La plate-forme Cormas est un environnement de développement des SMA pour les problèmes de dynamique et d'usage de ressources. Elle a été établie au Cirad qui est un institut de recherche en agronomie pour le développement au sein d'une équipe Green (Gestion des ressources renouvelables, environnement) qui concevait la gestion des ressources comme un problème d'interactions entre des dynamiques naturelles et des dynamiques sociales.

La plate-forme est plus qu'un prototype mais reste un produit évolutif. Le logiciel est libre et disponible sur un site ftp; l'équipe assure des formations d'une durée de deux semaines sur le thème des Sma et de la gestion des ressources. Cormas est élaborée grâce au langage Smalltalk et s'appuie sur le langage pour le développement de modèles. Utilisant l'environnement VisualWorks dont il existe une version non commerciale, Cormas peut être utilisé indifféremment sous Unix, Windows ou Mac. Les modèles développés sont donc directement transférables.

Plusieurs applications ont été développées (autour de quinze). Elles concernent toutes la simulation de société d'agents qui exploitent une ressource qui peut elle même être dynamique et simulée par un Sma **[4,5,6,7,8,9]**.

Modèles multi agents

Cormas est un environnement qui propose à l'utilisateur de se construire son modèle pour le traiter par simulation. Il propose donc un espace qui peut être structuré sur plusieurs échelles et des agents qui sont des boites assez vides, à charge pour l'utilisateur de construire son agent comme il le pense, en s'appuyant s'il le veut sur des modèles préexistants. Suivant le modèle (l'application) nous aurons des agents réactifs ou délibératifs, socialement contrôlés ou pas, raisonnant sur le temps ou pas, stratège ou opérationnel, etc. Ce sont les applications qui nous font construire ces agents. Dans un deuxième temps, en cas de généricité d'agents ou de protocoles, l'innovation est intégrée à la plate-forme.

Par défaut Cormas propose une représentation de l'espace type automate cellulaire, ainsi que l'implémentation d'entités spatiales qui sont des agents qui représentent des portions d'espace. Cormas propose ensuite trois types d'agents,

les agents spatialisés qui sont situés sur l'espace et pour lesquels Cormas propose des primitives de déplacement.

Les agents communicants qui possèdent une boîte aux lettres et qui sont connectés par un canal de communication. Cormas propose les primitives de communication.

La notion de Groupe est réifiée.

Toute entité de Cormas possède des programmes destinés à gérer le partage de ressources suivant différentes modalités (premier demandé, premier servi, etc.).

Enfin Cormas propose d'implémenter des objets situés ou non qui donnent au modélisateur la possibilité de représenter un environnement complet.

Après avoir représenté les entités du modèle, il s'agit de représenter le contrôle de la simulation. Le modélisateur programme alors une entité pour gérer l'activation des agents suivant un séquençage ou suivant différents modèles (événements discrets) ou protocoles.

Enfin, le modélisateur doit programmer son observation du monde à travers la notion de point de vue. Deux interfaces sont disponibles. L'une est spatiale et peut être associée à un Système d'Information Géographique, l'autre permet l'observation des liens sociaux des agents à travers leurs communications. Enfin la troisième interface pour le « monitoring » est constituée de classiques graphes scientifiques.

Il est possible de représenter jusqu'à 5000 agents si ceux-ci sont très réactifs.

Chapitre 3

Mimosa : Nouvelle approche dans la représentation de connaissances et la simulation [2]

La plateforme

L'objectif de Mimosa est de spécifier les outils génériques permettant aux modélisateurs de décrire leurs modèles ainsi que les simulations. La plupart des plateformes de modélisation et de simulation définissent les formalismes dans lesquels le modèle est exprimable ou laissent partiellement la liberté de définir ces formalismes, mais en donnant l'accès direct au langage de programmation sous-jacent. Le principe est le suivant :

- Un modèle est un discours sur l'expérience. Ce discours articule des concepts, que ces concepts réfèrent à des objets individuels (on parlera de concepts individuels) ou à des catégories d'objets individuels (on parlera alors de concepts catégoriels). Nous appellerons modèle ou point de vue un ensemble de concepts.
- Ces discours constituent autant de formalismes au sens large, c'est-à-dire pas mathématique tant que ces formalismes ont un caractère formel au sens qu'ils ont une forme, donc une syntaxe.

En conséquence, un outil générique de modélisation doit permettre de définir à la fois les moyens du discours (les formalismes) et le discours lui-même (les concepts individuels et catégoriels). La modélisation d'un système complexe, par exemple un écosystème, nécessite la multiplication des points de vue (écologique, agronomique, sociologique, économique…) qui sont autant de discours ayant chacun leur spécificité de par le formalisme utilisé et les concepts énoncés dans ce formalisme, mais qu'il s'agit d'articuler. En passant, nous insistons sur l'articulation plutôt que sur l'intégration qui supposerait ou imposerait un discours universel et ultime dans lequel tout pourrait s'exprimer. Cette articulation peut se faire à différents points selon des niveaux d'abstraction différents. Pour traiter ce dernier cas, nous invoquons l'approche holonique ou hiérarchique au sens de la théorie de la hiérarchie en écologie. Dans cette approche, il s'agit d'articuler le

point de vue dans lequel l'objet du discours est vu comme un tout en interaction avec son extérieur avec celui où l'objet est vu comme un composé d'un ensemble d'objets considérés à ce niveau et leur tour comme des « tout ». Ainsi tout Holon est à la fois un tout indécomposable vis-à-vis de l'extérieur et un composé dans son fonctionnement interne. Nous proposons d'aborder cet aspect à travers les notions génériques de composant (les « tout ») et de composés.

En conséquence, l'objectif de MIMOSA se décline en la mise en place d'outils permettant de spécifier les formalismes utilisés et les modèles exprimés dans ces formalismes en faisant l'hypothèse que tout formalisme (donc tout modèle) peut se décrire en termes de composants et de composés. Nous entendons donc à ce stade la généricité comme la possibilité d'être multi formalisme, multi modèle, (on dit aussi multi point de vue) et multi niveau. Cette généricité se veut issue d'une réflexion sur la modélisation, et plus généralement de représentation des connaissances dans la mesure où l'intelligence artificielle et l'informatique proposent également des formalismes de représentation autres que les formalismes strictement mathématiques.

Nous proposons donc une boîte à outil permettant de décrire une vaste gamme de formalismes, puis les modèles exprimés dans ces formalismes et de les articuler entre eux en vue de modéliser des systèmes complexes. Après un travail comparatif sur les différents formalismes utilisés dans la littérature, nous faisons l'hypothèse que cette boîte à outil peut être constituée de notions très simples permettant de décrire les structures et les processus, à savoir :

- Les composants, les composés et les relations pour les descriptions structurelles
- Les mesures, les évènements, les états et les fonctions de transition pour les descriptions dynamiques

Description des modèles

L'ensemble des formalismes sont exprimés à partir de notions de base élémentaires qui de ce fait n'ont pas de sémantique particulière. Ces notions sont divisées en deux catégories :

1. Les notions structurelles pour décrire les choses dont on veut parler
2. Les notions dynamiques pour décrire les processus

Les notions structurelles

Pour les notions structurelles, nous avons :

- la notion de composant pour décrire une entité non décomposable ;
- la notion de composé pour décrire un agrégat de composants. De plus, un composé décrit comment il nomme ses composants ;
- la notion de relation qui peut elle-même se décliner selon trois types de relation :

- les relations internes ou intra-composés qui décrivent des relations entre les composants d'un composé, donc qui sont définies entre les éléments d'un ensemble.
- Les relations inter-composés qui décrivent les relations entre deux composés, donc d'un ensemble dans un autre ;
- Les relations entre un composant et un composé : il en existe une par défaut qui est la relation d'appartenance.

Ces notions n'ont pas de sémantique en elles mêmes mais nous faisons l'hypothèse qu'elles peuvent servir de base pour décrire n'importe quelle type de représentation et donc de formalisme.

Prenons quelques exemples :

- En intelligence artificielle, les connaissances sont parfois représentées sous forme de concepts individuels ou catégoriels décrits par des attributs et relations entre concepts. On le retrouve sous une forme logique avec les logiques de description utilisées pour décrire des ontologies. Ce type de formalisme peut être mis à disposition en considérant que les composants sont les concepts élémentaires et les composés sont des regroupements de concepts élémentaires nommés par des noms d'attribut. Rappelons ici qu'un

composé définit également les noms pour accéder à ses composants qui sont ici des noms d'attribut.

- Les modèles à compartiments sont construits à partir de compartiments ou variables et de flux entre les compartiments. On peut aisément considérer que les composants sont les compartiments et les composés des ensembles de compartiments nommés par des noms de compartiment, c'est à dire les modèles à compartiments eux-mêmes. Les flux peuvent être représentés par des relations entre les compartiments. A ce stade, nous ne décrivons pas la dynamique du modèle mais seulement sa structure.
- Un espace peut être représenté comme un ensemble de lieux muni d'une topologie, par exemple sous la forme d'une relation de voisinage. Les composants sont alors les lieux, le composé est l'espace dont les composants sont nommés par des coordonnées ou des noms de lieu selon l'usage. La relation interne est la relation de voisinage. Les relations entre composés peuvent permettre de décrire les relations possibles entre deux espaces, notamment les changements de coordonnées.

Nous pouvons donc décrire ainsi les notions avec lesquelles il est possible de construire un discours sur l'espace, les concepts, les modèles dynamiques, le temps, etc. Il s'agit ici de définir les moyens du discours.

A partir de là, il devient possible de construire le discours proprement dit : si on a des formalismes pour décrire des espaces, on peut décrire des espaces, si on a des formalismes pour décrire des modèles à compartiments, on peut décrire des modèles à compartiments, etc. Dans le modèle, on distingue encore les types (les concepts catégoriels) et les instances (les concepts individuels). Les types décrivent des ensembles d'objets, alors que les instances décrivent des objets particuliers. Ainsi, on pourra parler des espaces discrets à deux dimensions munis d'un voisinage de VonNeumann qui sont donc décrits par un type, et on pourra parlé d'un espace discret à deux dimensions particulier qui est décrit par une instance.

Nous avons donc quatre niveaux de discours :

1. Le niveau asémantique ou boite à outils qui sert de base générique à la plateforme MIMOSA pour décrire les différents formalismes pertinents à

chaque modélisation. Ce niveau met à disposition les notions de composant, composé et relation qu'il faudra décliner dans le niveau suivant ;

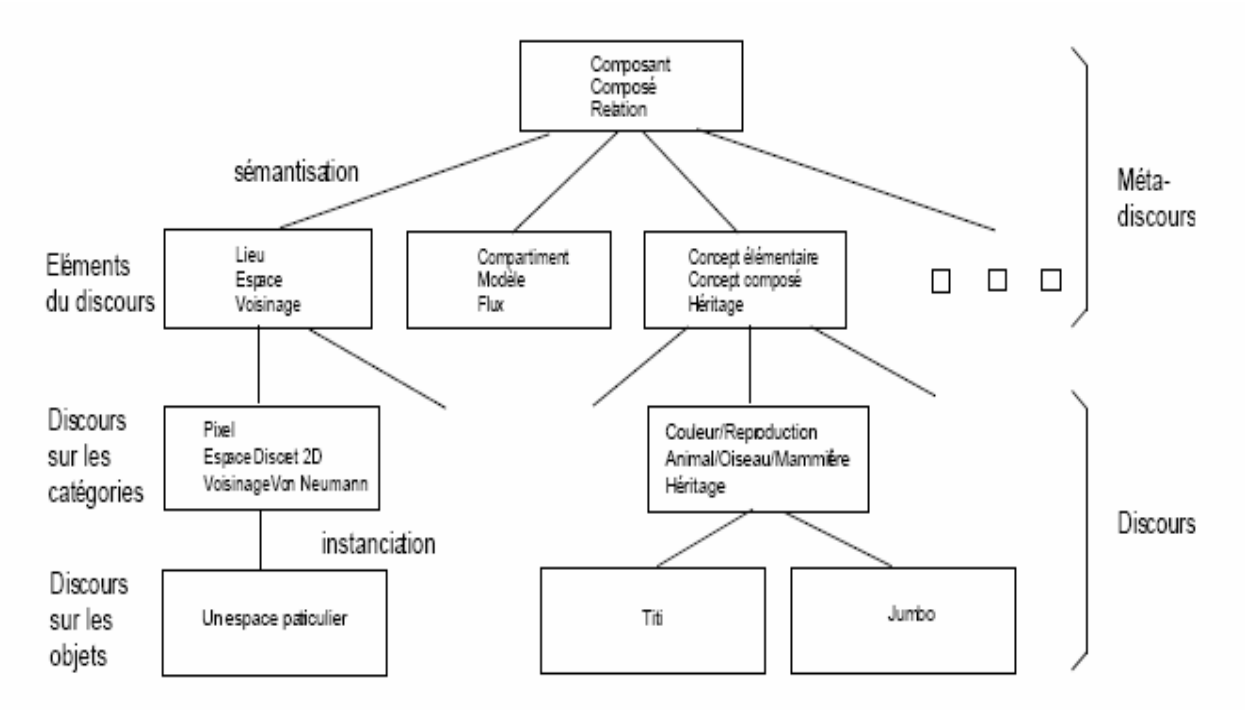

Figure 2: Architecture de MIMOSA

2. le niveau des formalismes qui est directement programmable dans la plateforme MIMOSA pour les différents besoins spécifiques. Un certain nombre d'éléments pour parler de l'espace, des concepts sont déjà implémentés mais d'autres peuvent être ajoutés par programmation au fur et à mesure des besoins. Ce niveau met à disposition ce que l'on appelle les méta concepts, puisqu'il s'agit du formalisme pour exprimer le modèle et non pas les modèles eux-mêmes
3. Le niveau des types ou des concepts catégoriels permet de définir les concepts catégoriels dont on va parler : les types de composé, les types de composant et les types de relation. Nous déjà au niveau du modèle proprement dit et donc ce niveau met à disposition les moyens pour le modélisateur de décrire son modèle.
4. Finalement, le modélisateur va utiliser les types qu'il s'est donné dans les niveaux précédents pour construire un ou plusieurs modèles concrets dont il va pouvoir exécuter la simulation

Les deux premiers niveaux sont accessibles par programmation (en l'occurrence Java), le premier niveau étant figé une fois pour toute, le deuxième proposant un ensemble de formalismes prédéfinis et mécanisme d'extension pour décrire les différents formalismes que l'on veut pouvoir utiliser. Les deux suivants sont accessibles aux modélisateurs via une interface graphique dédiée. Cette interface doit fournir à terme les moyens de construire des modèles indépendamment du langage d'implémentation et dans le ou les formalismes choisis par le modélisateur.

L'implémentation est construite à partir de classes abstraites Java suivantes :

ComponentType : la classe qui correspond aux parties des formalismes qui dénotent des concepts catégoriels composants appelés aussi types de composants

CompoundType : la classe qui correspond aux parties des formalismes qui dénotent des concepts catégoriels composés appelés aussi types de composés

Component : la classe qui correspond aux parties des formalismes qui dénotent des concepts individuels unitaires, appelés aussi les instances de composants ou simplement composants

Compound : la classe qui correspond aux parties des formalismes qui dénotent des concepts individuels composés, appelés aussi instances de composés ou simplement composés.

Name : une interface dont les implémentations doivent définir ce qu'est un nom pour le composé

A l'exception de Component, toutes les classes possèdent un nom. Ce n'est pas le cas des composants parce que nous considérons que le nom n'est rien d'autre que ce qui permet de le désigner dans le composé. On peut dire qu'il en possède un donc mais il n'est pas de même nature. Les concepts individuels se veulent explicitement des instances des concepts catégoriels et mettent donc à disposition le type du concept catégoriel correspondant.

Chaque composant définit le moyen d'accéder à son composé, aux autres composants de son composé, ainsi qu'aux composants et composés avec lesquels il est mis en relation

Chaque composé permet d'accéder, de détruire, de modifier, d'ajouter ses composants.

Sur ces ensembles de base, on définit également un ensemble de types de relations :

IntraCompoundRelationType : la classe qui correspond aux parties des formalismes qui dénotent des types de relation entre composants à l'intérieur d'un même composé

InterCompoundRelationType : la classe qui correspond aux parties des formalismes qui dénotent des types de relation entre composants de composés distincts.

ComponentCompoundRelationType : la classe qui correspond aux parties des formalismes qui dénotent des types de relation entre un composant et un composé

Ainsi que leurs instances respectives : IntraCompoundRelation, InterCompoundRelation, ComponentCompoundRelation. Ces notions sont également munies d'un nom et surtout d'une fonction qui permet de transformer un élément dans un ou plusieurs autres. Cette fonction dépend également du formalisme qui lui donne un sens (par exemple, une transformation de coordonnées dans les formalismes sur l'espace).

Les notions dynamiques

Les notions suivantes sont mises à disposition pour décrire les processus :

- Les états sont associés indifféremment aux composants et aux composés et permettent de décrire l'état dynamique
- Les évènements induisent des changements d'état. A ce titre il sont traités et générés par les composants et les composés.
- Les mesures permettent d'obtenir des informations sur l'état actuel des composants et des composés. Pour les composés, la mesure minimale est l'accès aux composants.
- Les fonctions de transition permettent de décrire comment un composant ou un composé réagit aux évènements en changeant éventuellement d'état et/ou en engendrant des nouveaux évènements.

Encore une fois ces notions n'ont pas de sémantique. Il est nécessaire pour chaque formalisme de décrire ce qu'est un état, un évènement, des mesures, et comment se

décrit la fonction de transition en fonction des besoins. Pour illustrer l'utilisabilité de ces concepts, citons les exemples suivants

- Les états peuvent être ceux d'un automate d'état fini (Q), les évènements comme un alphabet quelconque (E), la fonction de transition comme une table : Q x E →Q ainsi que la fonction de sortie Q→E
- Les états d'un composé sont en général l'ensemble de ses composants à un moment donné, les évènements l'ajout et le retrait d'un composant. On peut aisément les généraliser à diverses agrégations calculées à partir des états des composants. Comme nous l'avons suggéré, une mesure possible peut être l'accès à un composant mais on peut en imaginer d'autres comme la cardinalité, la compacité, etc.
- Pour la cellule (un composant) d'un automate cellulaire (un composé) du jeu de la vie, l'état peut avoir les valeurs « vivant » ou « mort », la mesure est l'accès à cet état et la fonction de transition est la règle bien connue du jeu de la vie.

Nous retrouverons ainsi les deux niveaux des notions structurelles à savoir une boîte à outil asémantique et un niveau de spécification des formalismes nécessaires. Encore une fois, un ensemble de formalismes sont déjà prédéfinis dans le système auquel s'ajoute un mécanisme d'extension.

Nous avons choisi d'accrocher systématiquement les descriptions des processus aux descriptions structurelles. On associe donc à chaque concept catégoriel qu'il soit de l'ordre du composant ou du composé, l'ensemble des évènements et mesures qu'il définit ainsi que l'état et la fonction de transition.

L'espace

Nous ne supposons pas qu'il existe un espace objectif qui serait euclidien, continu et tridimensionnel. Les mathématiques définissent une multitude d'espaces distincts, et l'espace dans le langage commun est également décrit de façon multiple et souvent incommensurable. Afin de permettre au modélisateur de décrire l'espace qui convient le mieux à ses besoins, nous avons introduit les notions qui suivent.

Un espace est un ensemble de lieux (champs, villes, villages, cours d'eau…). Nous introduisons donc l'espace comme un composé (Space) et le lieu comme un

composant (Place). Les lieux à l'intérieur d'un espace sont désignés par des noms et parfois, mais pas nécessairement toujours, situés les uns par rapport aux autres par des relations définies sur un composé (IntraCompoundRelationType) suffisent aux besoins. C'est le rôle du modélisateur de se donner la batterie de relation qui lui convient.

A noter que nous distinguons la notion de lieu de l'objet géométrique qui peut en servir de représentation dans une cartographie. Le discours sur les objets géométriques fait partie d'un autre formalisme adapté au discours géométrique que l'on peut ensuite articuler au discours sur l'espace à l'aide de relations entre composés (InterCompoundRelationType). Il servira ultérieurement à engendrer les visualisations graphiques. Il faudra également prévoir la possibilité de calculer automatiquement certaines relations qualitatives (adjacence, accessibilité…) à partir de leurs représentations géométriques.

Un pas plus loin, nous définissons la notion d'espace métrique (MetricSpace) qui est également un composé constitué de lieux. Toutefois, les lieux sont désignés par des coordonnées plutôt que par des noms de lieux, ou, plus exactement, les coordonnées tiennent lieu de nom dans ces composés là.

Pour permettre au modélisateur de définir les espaces dont il a besoin, qu'ils soient discrets ou continus, mono, bi ou tridimensionnels, une batterie de classes sont définies pour permettre de les décrire : Discrete1DSpaceType, Discrete2DSpaceType, Discrete3DSpaceType ainsi que les noms de lieux qui leur correspondent. A partir de là, il devient facile de définir un ensemble de relations :

- de voisinage : sous la forme de relations sur un composé
- de contenance : sous la forme de relations entre des espaces distincts représentant des échelles ou des compositions spatiales différentes
- de transformation : d'un système de coordonnées dans un autre
- etc.

Les lieux sont purement descriptifs et ne sont, a priori, le support d'aucune dynamique. Par contre, ils peuvent contenir une variété d'attributs mesurables par des instruments de mesure (MeasureTool) éventuellement paramétrés par des coordonnées continues pour définir des champs sur chaque lieu.

En ce qui concerne les espaces, l'ensemble des lieux et des relations entre eux peuvent changer au fil du temps. Toutefois, à ce jour, cette dynamique n'a pas encore été implémentée sous forme de réponse à des évènements. Il est également possible de définir des attributs ou des champs mesurables sur tout l'espace.

Les automates cellulaires

Les automates cellulaires sont considérés comme des composés formés de cellules sur lesquelles est définie une fonction de voisinage. Très concrètement nous avons spécialisé les notions de lieux et d'espaces métriques discrets pour définir les automates cellulaires. Ainsi une cellule est un lieu muni d'un comportement dynamique et un automate est un espace mono, bi ou tridimensionnel constitué de cellules.

Les cellules répondent aux évènements suivants :

InitializeEvent : pour mettre la cellule dans son état initial

AsynchronousStep: pour mettre à jour directement l'état de la cellule avec le résultat du calcul de l'état suivant

SynchronousStep: pour calculer l'état suivant de la cellule sans modifier l'état courant

SynchronousUpdate : pour mettre à jour l'état de la cellule avec l'éat suivant

Les deux derniers évènements implémentent la technique du « double buffering ». L'initialisation et le calcul de l'état suivant est délégué à une implémentation de l'interface (CellBeahaviour) qui, à ce jour, doit être implémenté en Java pour chaque comportement souhaité.

Les automates répondent aux évènements suivants :

InitializeEvent : pour initialiser toutes les cellules

AsynchronousStep : pour faire évoluer l'ensemble des cellules de façon asynchrone

SynchronousStep : pour faire évoluer l'ensemble des cellules de façon synchrone

Les schémas conceptuels

Il s'agit de mettre à disposition le formalisme des schémas conceptuels pour permettre de décrire les concepts catégoriels et individuels ainsi que leur comportement. Ils permettent de construire facilement des ontologies.

Les schémas conceptuels catégoriels décrivent ce qu'il y a de commun entre un ensemble d'objets du discours ; Par exemple, un éléphant est gris avec deux défenses et de grandes oreilles. On peut organiser ces schémas conceptuels en hiérarchies avec des relations de super-type à sous-type. Un sous-type respecte toutes les descriptions de son super-type plus les siennes propres. A ce titre, un sous-type est plus spécifique que son super-type dans la mesure où moins d'individus correspondent à sa description.

A l'extrême, un schéma conceptuel ne décrit qu'un et un seul objet du discours auquel cas on parle de schéma conceptuel individuel. Il ne s'agit pas de la notion d'instance des langages orientés objets dans le sens que les descriptions des langages évoluent avec les description de la hiérarchie des super-types qui lui correspondent, alors que les variables d'instance dans les langages orientés objets prennent des valeurs quelconques et indépendantes des définitions initiales. La notion de variables d'instances au sens orienté objet est reprise par la notion d'état.

Nous faisons ainsi une synthèse entre les langages orientés objets et les schémas conceptuels (appelés aussi parfois représentations centrées objets par opposition aux représentations logiques où les objets du discours sont dispersés comme des constantes ou des variables dans les formules).

Nous avons introduit ce formalisme très général comme base de description pour la plupart des discours que le modélisateur souhaitera tenir. Pour cela un schéma conceptuel est un composé d'attributs, les composants étant des descriptions d'attribut. Les composants sont nommés par des noms d'attribut. Chaque attribut est décrit par son type et sa cardinalité. Par exemple, pour l'éléphant, l'attribut de nom « patte » peut être de type de composant « patte » et de cardinalité quatre. Les deux classes suivantes implémentent la notion de schéma conceptuel :

Schéma: le composé dont les composants sont des attributs repérés par des noms d'attribut (StringName), il peut y avoir plusieurs attributs de même nom selon la cardinalité.

Attribute : le composant d'un schéma

Il ne s'agit que d'une implémentation particulière de la notion de schéma. On pourrait en ajouter d'autres, par exemple qui interfacerait avec le schéma relationnel d'une base de donnée et dont les instances (schémas conceptuels individuels) correspondraient alors au n-uplets de la base de donnée.

Les populations

Dans les simulations, il est souvent question de populations d'individus. C'est pourquoi nous avons mis à disposition cette notion. Les composants sont les individus et les composés sont les populations. Chaque composant (individu) est nommé par un identificateur unique arbitraire (nous avons choisi un entier).

Les individus peuvent être munis de comportements arbitraires. On met à leur disposition la possibilité d'accéder à n'importe quel autre individu de la population par son identificateur ainsi qu'à l'ensemble des individus. Bien évidemment, il est toujours possible de définir sur les individus des relations quelconques et de les utiliser pour spécifier leurs comportements.

Les populations mettent à disposition la possibilité d'ajouter ou de retirer des individus de la population. Il est également possible de mesurer la taille de la population. Ces fonctionnalités sont mises à disposition via les évènements et les instruments de mesure suivants :

PopulationEvent : la classe des évènements qui permettent de modifier la population

PopulationMeasure : la classe des instruments de mesure qui permettent d'obtenir des informations sur la population.

Le temps

Quand on passe de la modélisation à la simulation, s'ajoute un discours sur le temps. Là, également, nous faisons l'hypothèse qu'il n'existe pas de temps en soi mais seulement des discours sur le temps (cyclique, subjectif, irreversible ou non, continu ou discret), chacun d'entre eux étant une mise en cohérence d'expériences distinctes du temps. En conséquence, nous avons choisi d'imposer au modélisateur d'exprimer explicitement son discours sur le temps et de s'en servir pour effectuer des simulations. Des modèles différents reposeront donc éventuellement sur des

modèles du temps distincts qu'il s'agira également d'articuler pour pouvoir simuler un système complexe.

Il est également important pour permettre la simulation de distinguer entre le discours sur le temps et le discours sur la génération du temps. Très rapidement dit, notre expérience première sur laquelle se construit notre discours sur le temps est basée sur les notions duales d'évènement et d'état. Nous vivons individuellement une succession d'évènements (qui délimitent autant d'états), ou une succession d'états (dont les transitions constituent autant d'évènements). Sur cette seule base, on ne peut que séquencer ces évènements et états. En particulier, il n'est pas possible d'y introduire la notion de date, ni de durée, encore moins d'un temps indépendant sur lequel ces évènements et états s'inscriraient. C'est cette expérience première que nous mettons à disposition dans notre boîte à outils puisque nous ne décrivons que les états, les évènements et les transitions. Si nous envoyons un évènement à un objet, cet objet, en fonction de sa fonction de transition, va générer d'autres évènements vers d'autres objets, générant une cascade de réactions faisant se succéder évènements et états à travers l'ensemble du modèle. Il ne s'agit ni plus ni moins que d'un calcul (au sens informatique) et nous engendrons donc ce que nous appellerons ***temps de calcul.***

Pour introduire la notion de date et de durée, il nous faut donc introduire des horloges qui produisent également des évènements et des états mais sont considérés comme références pour les autres évènements et états. C'est une façon naturelle pour passer d'un temps ordinal à un temps cardinal. Dans notre cas, c'est le moyen que nous avons choisi pour passer du temps du calcul au ***temps simulé***

La notion de temps indépendant que les horloges mesureraient (alors qu'elles en produisent un !) et sur lequel se placeraient les évènements et les états se trouve être une abstraction commode pour articuler des successions différentes selon le point de vue que l'on prend en compte pour construire son temps ordinal. Il est donc important qu'on puisse le mettre à disposition.

C'est ensuite un choix que ce temps soit cyclique ou linéaire, discret ou continu selon les besoins et les facilités d'expression. Ceci constitue un discours sur le temps dans le sens où les évènements et états sont donnés et qu'il s'agit de décrire les articulations entre eux. Ce discours est essentiel dans la modélisation car c'est dans ce discours qu'est représenté ce qui est produit par les processus, qu'il soit réels ou

simulés. C'est pourquoi nous proposons également que MIMOSA mette à disposition la possibilité de décrire à la fois les moyens d'un discours sur le temps et le discours lui-même.

Nous pourrions en rester là si on décrivait un seul processus, ce qui n'est pas le cas. Si nous avons au moins deux processus qui interagissent, il ne suffit pas de décrire comment ils produisent des évènements mais aussi comment l'observateur produit son discours sur le temps de ces processus pour pouvoir les articuler entre eux. Le relais par le discours est ici indispensable car il s'agit néanmoins de produire des processus compatibles avec le discours déjà construit par ailleurs sur une « réalité». Si ce n'était pas le cas, il ne s'agirait pas de la simulation de quelque chose mais de la production d'une autre réalité mais virtuelle celle là.

- Le discours sur le temps

  Pour construire un discours sur le temps, nous utilisons principalement la métaphore spatiale (essentiellement parce que nous représentons le déroulement d'un état au fil du temps par une courbe sur une feuille de dessin).

  Il n'est donc pas étonnant que le discours sur le temps soit similaire au discours sur l'espace. Ainsi un temps est un composé constitué d'instants ou d'intervalles qui ont des noms qui sont soit des mots, des dates pour les instants, des coordonnées temporelles pour les intervalles.

- Les horloges

  Pour transformer le temps du calcul dans le temps simulé, il nous faut des horloges.

  A ce jour, nous avons mis à disposition deux compartiments possibles des composants :

  ClockImplementation: qui réagit aux évènements *Initialize* pour mettre l'horloge à *zero* et *Advance* pour faire générer un certain nombre de « *top* » et faire avancer l'horloge.

  ClockedImplementation: qui reçoit un *top* et envoie un message donné aux composants ou composés qui lui sont liés par une relation « *Control* »

Chapitre 4

Interface Mimosa/SIG

Qu'est-ce qu'un Système d'Informations Géographiques

Un SIG est un système d'information intégrant des données localisées. Mais avant tout il répond à une problématique :

- Améliorer et simplifier la communication
- Organiser des connaissances thématiques
- Gérer le partage de l'information
- Prévoir, simuler

Ces différents points soulevés marquent le caractère incontournable des SIG dans le monde contemporain ou l'échange, le partage et l'accès immédiat à l'information voulue sont d'une importance capitale.

Par soucis d'un échange facile des informations géographiques entre différentes entités, un standard est en train d'être mis en place. La norme OpenGis est donc apparue avec un certain formalisme au niveau des objets géographiques qu'on tente de faire adopter par tous les concepteurs de SIG.

La norme OpenGIS

Aujourd'hui, dans un environnement de plus en plus interconnecté, la tendance est au partage de l'information et des ressources pour leur traitement : on se dirige résolument vers des systèmes distribués et ouverts *(middleware / open distributed component-based systems)*. Les moyens pour y arriver constituent les fondements de l'OpenGIS.

Le développement des concepts et des technologies de l'OpenGIS est réalisé et coordonné par l'OpenGIS Consortium (OGC), regroupant des organismes privés et publics actifs dans le domaine des technologies de l'information géographique.

La vision de l'OGC est la pleine intégration des données géographiques *(geospatial data)*, et des ressources pour leur exploitation *(geoprocessing resources)*, dans le courant des technologies de l'information, avec une utilisation

large de systèmes géo informatiques commerciaux et interopérables au sein de l'infrastructure globale de l'information [5]

Au delà de cela la norme OpenGIS répond à un certain nombre d'objectifs techniques :

- Fournir une définition univoque des types de primitives géométriques.
- Supporter la dimension temporelle.
- Etre en accord, dans la mesure du possible, avec les standards existants dans le domaine de l'information géographique.
- Etre indépendant des systèmes d'exploitation, des langages de programmation, des interfaces homme-machine, du matériel, et des réseaux.

- Opérer sur toutes les plates-formes informatiques DCPs *(Distributed Computin Platforms)* (COM, CORBA, Java, etc.).
  - Opérer avec les principaux langages de bases de données (SQL, etc.).

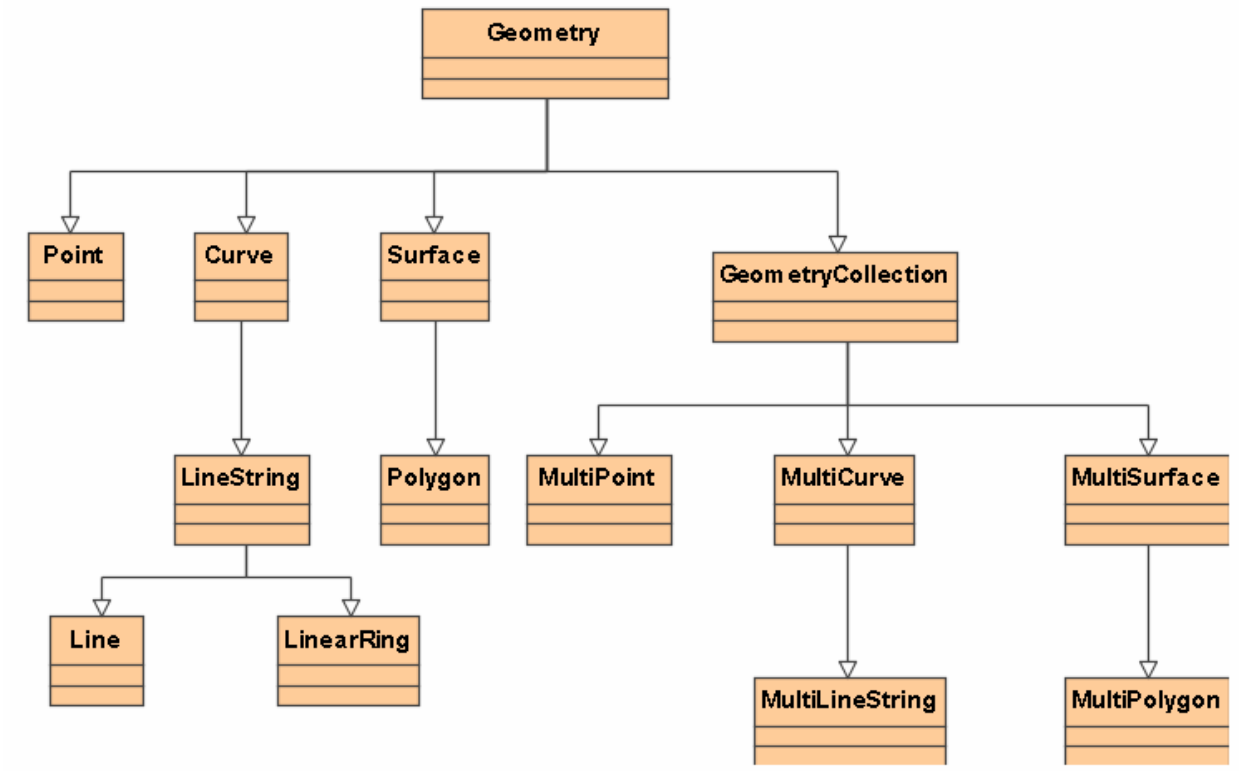

Figure 3:Relation d'héritage entre classes du modèle géométrique OpenGIS

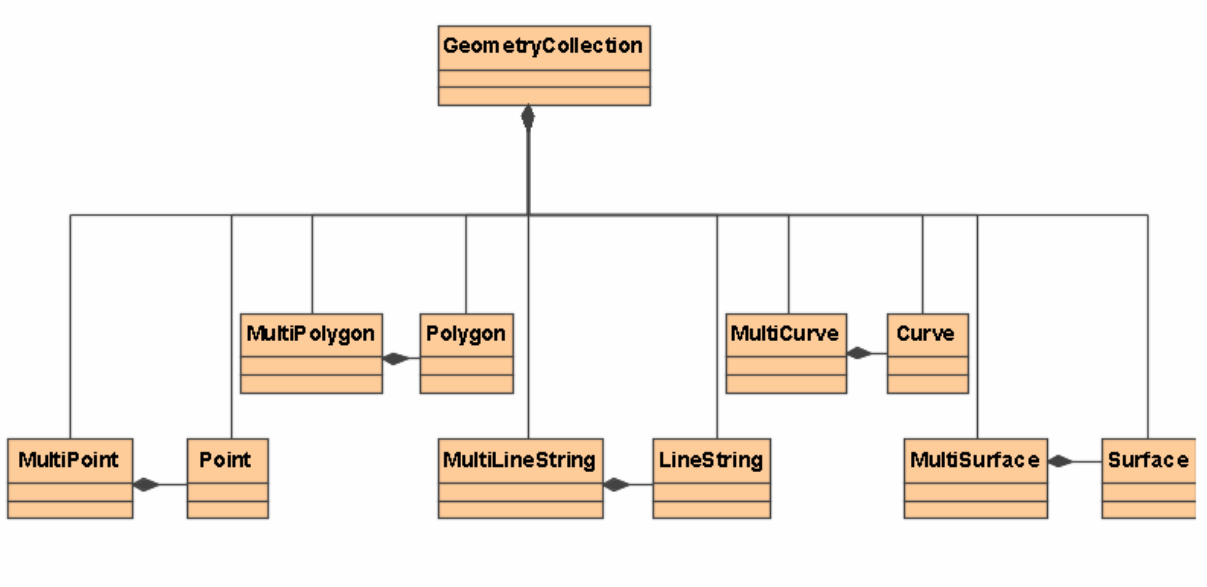

Figure 4: Relation de composition entre classes Mimosa/SIG

Pourquoi interfacer Mimosa avec des données géographiques?

Les chercheurs et les experts qui sont sensés utiliser Mimosa ont besoin d'un tel outil pour décrire de façon simple les concepts ayant trait à leur domaine d'évolution. Dans le cadre de la gestion de l'environnement et des territoires, ces concepts introduisent la nécessité de disposer d'objets géographiques permettant de décrire fidèlement le modèle en question. Une base de données SIG permettra alors une gestion efficace de ces objets, un partage adéquat de l'information et une facilité d'accès remarquable à celle-ci.

Qu'est-ce qu'il s'agit de faire ?

Globalement il s'agit de développer une interface permettant d'utiliser les différentes fonctionnalités d'un SIG dans les modèles élaborés avec la plateforme MIMOSA. Et cela passe nécessairement par une représentation des données de la base de données SIG au niveau de MIMOSA. Les données d'un SIG sont principalement de deux types :

- Des données géométriques
- Des données attributaires

Les données géométriques : pourquoi les représenter au niveau de Mimosa ?

Une raison évidente de cette représentation est la visualisation des environnements des systèmes qu'il faut modéliser.

En effet, pour visualiser l'environnement dans lequel baigne un modèle, la carte d'une région par exemple, la représentation géométrique de celle-ci est nécessaire.

Les différentes API de visualisation de modèles utilisées par Mimosa, doivent pouvoir disposer de données géométriques de base de l'élément à visualiser.

Les données attributaires

Les données attributaires sont les données qui accompagnent celles géométriques à des fins d'illustration. Par exemple : Nom de la ville, Nombre d'habitants etc.

Ces données doivent être représentées au niveau de Mimosa pour permettre au modélisateur de voir les données qui sont en relation avec les figures géométriques qu'il visualise.

Description d'un modèle SIG

Un modèle SIG comme tout autre modèle MIMOSA est décrit à travers la définition de composants, de composés et de relations. L'édition d'un modèle SIG passe donc par la définition claire des éléments sémantiquement parlant pour le modèle en question, en tant que Composants, Composés ou Relations.

Les composants

Un SIG peut être considéré comme un ensemble de couches qui représentent des entités indécomposables. Par exemple, pour un système d'information géographique d'un pays, on peut avoir différentes couches telles que celle des villes, des routes, des cours d'eau etc. Ces différentes couches, vues ainsi, peuvent être considérées comme des composants.

Nous définissons ainsi les classes suivantes :

LayerType : qui représente le composant type correspondant

LayerView : la vue correspondante au composant type LayerType

Les composés

Le SIG en lui-même peut être vu comme un tout, un ensemble de couches, de composants, donc un composé.

Nous définissons ainsi les classes suivantes :

GISType : qui représente le composé type correspondant

GISView : la vue correspondante au composé type GISType

GISType définit un ensemble de méthodes permettent de manipuler ses différents composants (getLayerType(), addLayerType(), getLayerType[](), …)

Les relations

Un modèle SIG peut servir d'environnement pour un modèle Système Multi Agents et cela passe par l'établissement de relations entre ces deux modèles. On peut donc définir une relation inter-composés entre un groupe d'individus (une population) et un espace particulier (un SIG).

Chapitre 5

Edition d'un modèle SIG

Le composé type : GISType

On crée un composé type de type **GISType** qui correspond aux Systèmes d'Informations Géographiques. Cela passe par la saisie des éléments de connexion à la base de données géographiques sous-jacente au modèle que l'on veut créer.

Ainsi, on renseigne les éléments suivants :

- User : L'utilisateur concerné par la connexion courante
- Password : son mot de passe
- Server : l'adresse IP du serveur postgresql
- Port : le port concerné au niveau du serveur
- Database : la base de données du Système d'Informations Géographiques en question

Remarque : Il est important de bien spécifier la classe qui implémente le Composé type défini, en l'occurrence ici mimosa.gis.GISType.

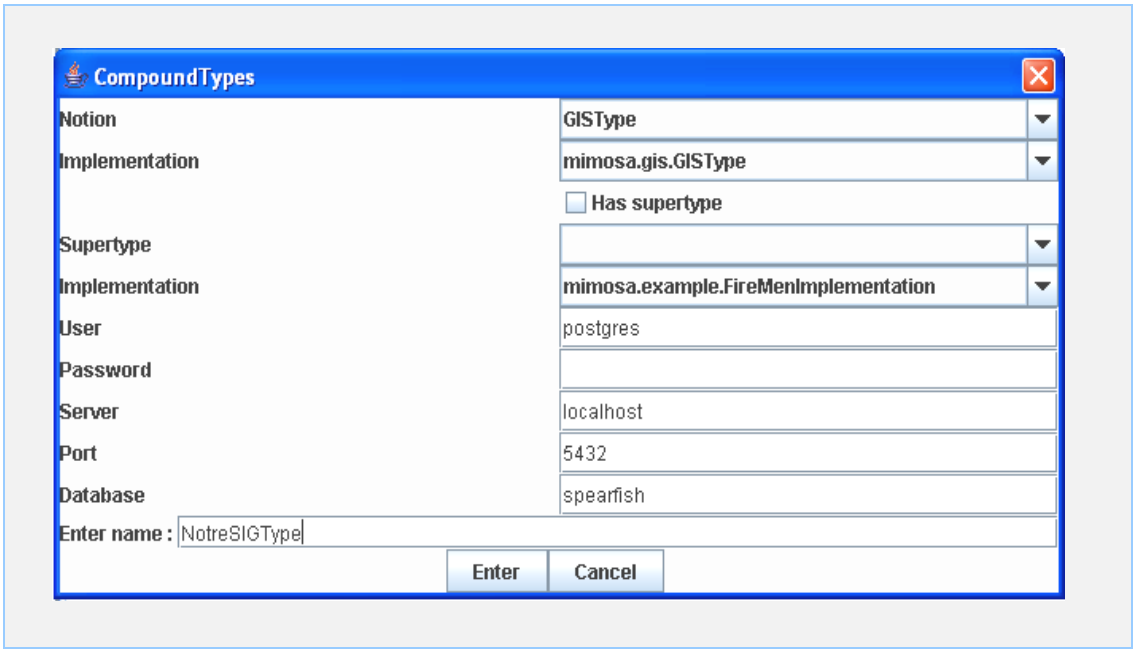

Figure 5:Définition d'un composé de type GISType

Le composant type : LayerType

Les composants types définis sont liés à la notion LayerType qui correspond plus explicitement à ce que l'on peut nommer « types de couches ».

Pour définir un nouveau composant type LayerType, on procède de la manière suivante :

- On ouvre l'onglet correspondant **component types**
- Click droit, puis choisir le menu **add component**
- Une nouvelle fenêtre s'affiche, choisir la notion **LayerType**
- Cocher la ou les tables correspondantes à la nouvelle couche type
- Donner un nom au nouveau composant type et valider.

Remarque : Veillez à bien préciser le GISType correspondant, NotreSIGType ici.

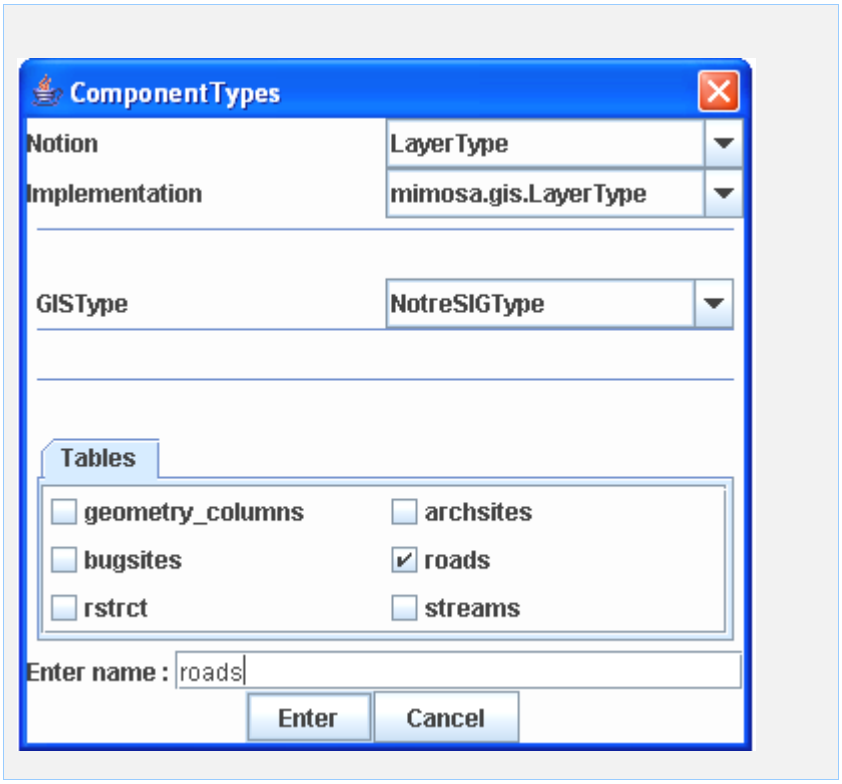

Figure 6: Définition d'un composant type de type LayerType

Le modèle

La création du modèle relève d'une instanciation des composés et composants type précédemment défini Pour créer le modèle SIG donc, on procède de la manière suivante :

Etape 1 : création du modèle

- On ouvre l'onglet correspondant **Model**
- Click droit, puis choisir le menu **add Model**
- Une nouvelle fenêtre s'affiche, donner un nom au nouveau modèle

Figure 7: Création d'un modèle

Etape 2 : Instanciation du composé type correspondant

- On clique sur le modèle que l'on vient de créer, monModeleSIG ici
- La fenêtre de description du modèle s'affiche en dessous
- Click droit, puis choisir le menu **add Compound**
- Une nouvelle fenêtre s'affiche, choisir la notion GIS et donner un nom au nouveau composé

Remarque : Veillez à bien préciser la notion et le composé type correspondant, respectivement ici, GIS et NotreSIGType.

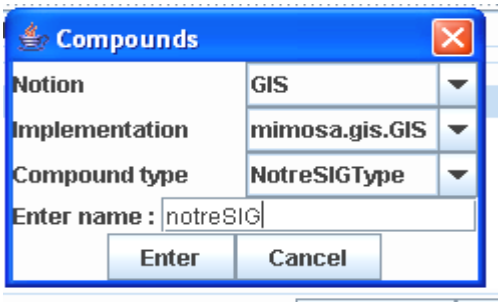

Figure 8:Instanciation d'un composé de type GISType

Chapitre 6

Application aux Systèmes Multi Agents

Les agents

Il s'agit de définir des individus et des groupes d'individus que l'on peut considérer comme des agents et des groupes d'agents.

Etape 1 : Définir un individu type

- On ouvre l'onglet correspondant **component types**
- Click droit, puis choisir le menu **add component**
- Une nouvelle fenêtre s'affiche, choisir la notion **Individual Type**
- Et créer un nouveau type d'individu

Etape 2 : Définir une population type

- On ouvre l'onglet correspondant **compound types**
- Click droit, puis choisir le menu **add compound**
- Une nouvelle fenêtre s'affiche, choisir la notion **Population Type**
- Choisir l'individu type correspondant
- Donner le nom de la nouvelle population type et valider

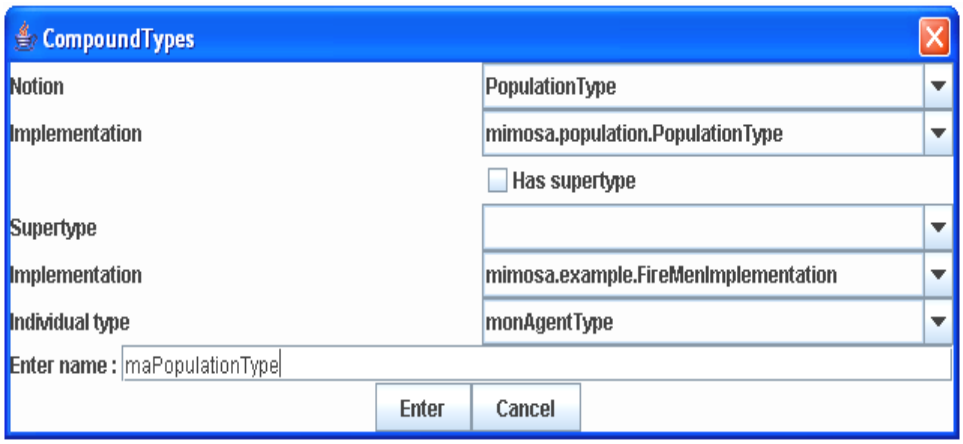

Figure 9: Définition d'un compsé type de type PopulationType

L'environnement

L'idée ici, est de permettre aux modélisateurs de positionner des agents sur un environnement de type SIG. Il s'agit donc de créer les composés et composants nécessaires à la mise en oeuvre de cet environnement.

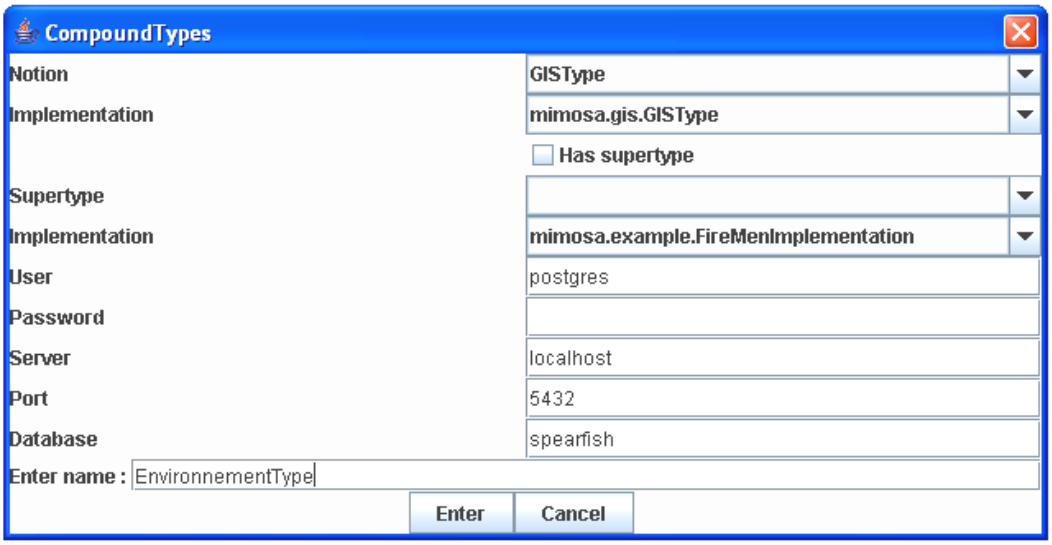

Figure 10: Définition du composé type EnvironnementType

On crée ici le Composé type **EnvironnementType**

Et les composants type correspondant…

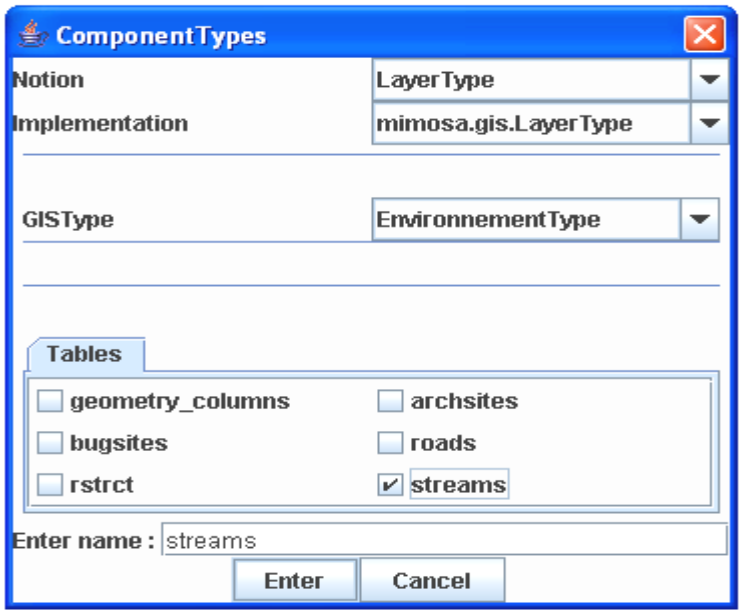

Figure 11: Définition du composant type Streams

La relation Agents – Environnement

On doit associer le groupe d'agent défini à un type d'environnement pour pouvoir constituer de manière complète le Système Multi Agents. Il s'agit donc de définir une relation inter composés entre les composés type **PopulationType** et **EnvironnementType**.

- On ouvre l'onglet correspondant **Relation types**
- Click droit, puis choisir le menu **add relation type**
- Une nouvelle fenêtre s'affiche, choisir la notion **InterCompoundRelationType**
- Choisir les composés type PopulationType et EnvironnmentType comme

Group type.

- On valide pour créer la nouvelle relation type

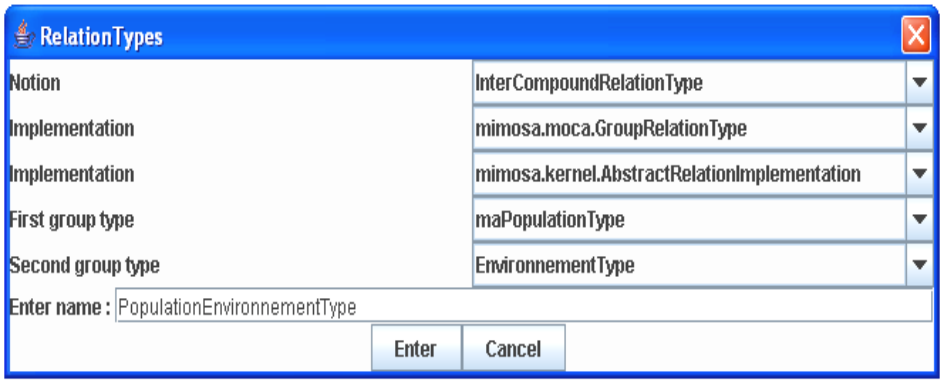

Figure 12:Définition de la relation type PopulationEnvironnementType

Remarque : L'ordre dans lequel on spécifie les group type (first ou second) n'est pas important ici.

Création et visualisation du modèle

On crée le modèle tel que spécifié précédemment

- On ajoute un composé de type SIG pour définir l'environnement du SMA
- On rajoute successivement les composés représentant les différentes populations d'agents que l'on veut positionner dans l'environnement du SMA.
- Pour visualiser le modèle, on clique deux fois sur le composé de type SIG.

Un exemple précis

Ici nous choisissons de créer un modèle sur fond d'un Système d'Informations Géographiques des Etats-Unis. Nous procédons de la manière suivante :

- On crée deux types de populations d'agents (compoundType) :
  - Les américains
  - Les non américains

- On crée l'environnement de type SIG que l'on nomme **Etats-Unis** (compoundType).

- Puis l'on crée le modèle en instanciant les différents types proposés précédemment, on définit ainsi :

- Deux groupes d'américains : les noirs américains et les blancs (composés)
  - Un groupe de non américains : les non américains (composé)
  - L'environnement qui est une instanciation du composé type Etats-Unis

- On double clique sur le composé SIG, pour visualiser le modèle

On obtient le résultat suivant :

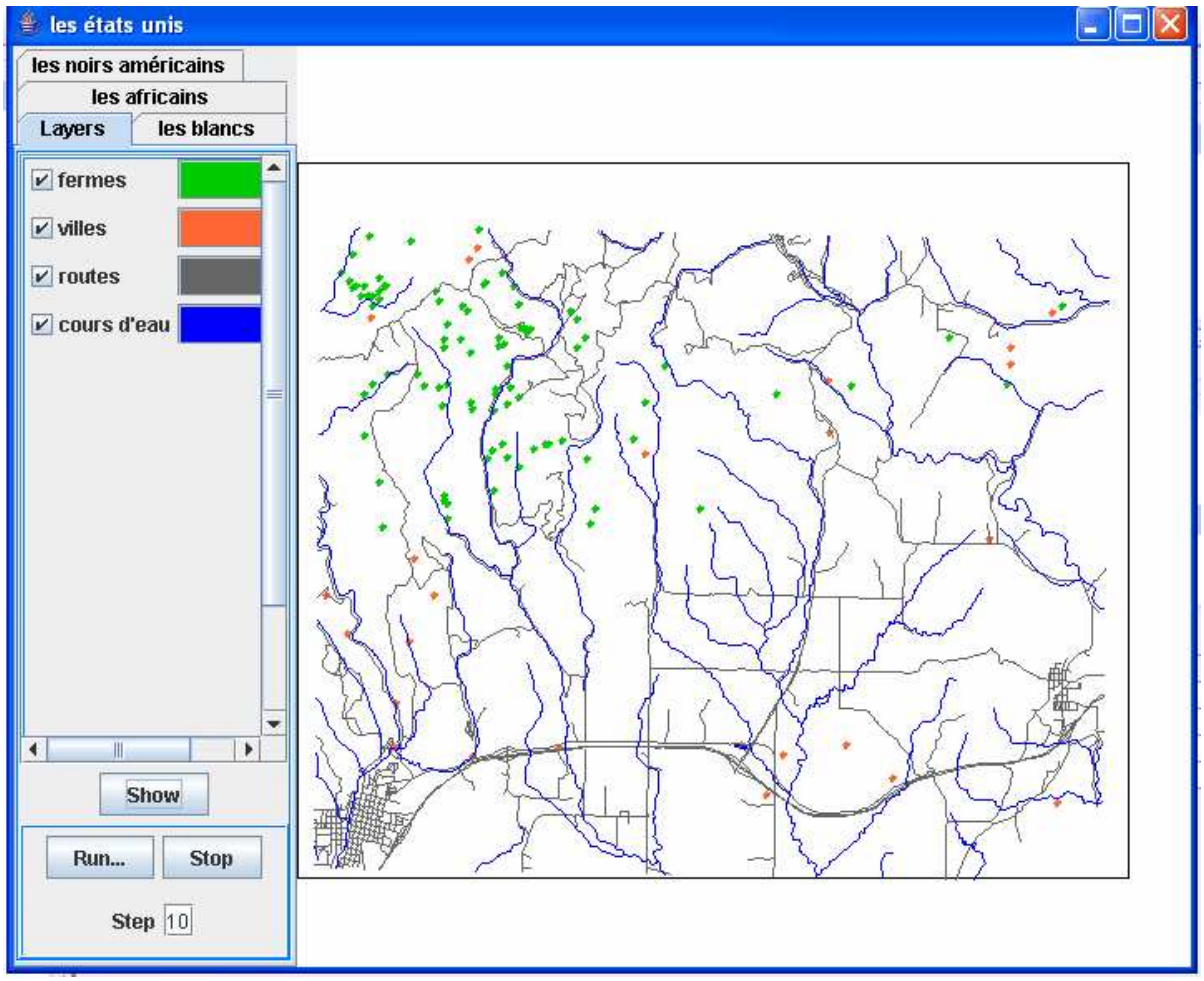

Figure 13:Visualisation du modèle SIG

On peut visualiser les différentes populations d'agents, par la suite, en leur attribuant une représentation particulière.

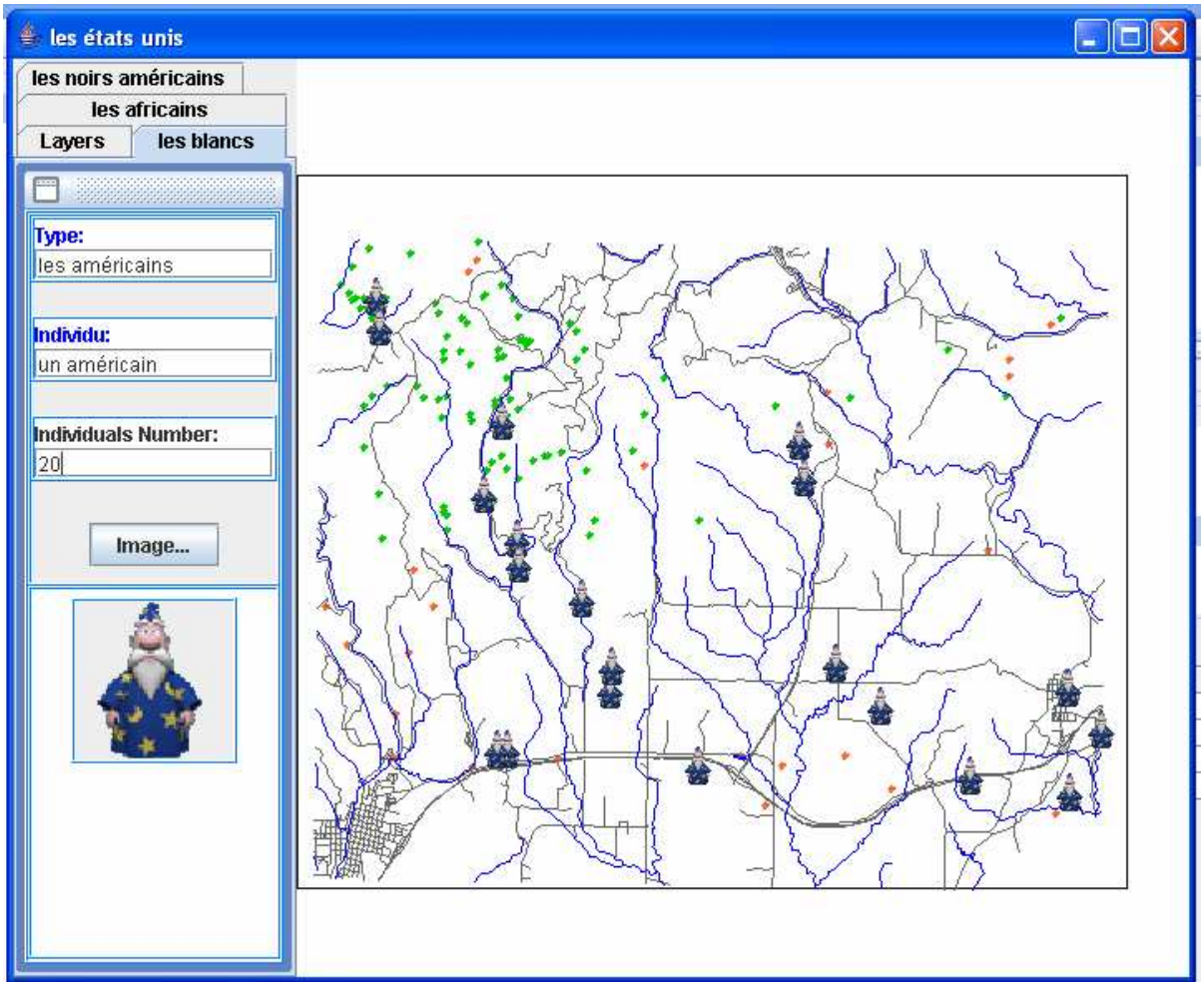

Figure 14: Visualisation des agents

Table des figures

Table d'index

R

S

T

V

Bibliographie

[1] Swarm

www.swarm.org

[2] Jean Pierre MULLER

Mimosa : représentation et simulation des connaissances

20 septembre 2004

[3] Jacques Ferber

www.madkit.org

[4]LIEURAIN E., " Couplage Cormas - Access - ArcView. Une étude de faisabilité à travers la construction d'un modèle de parcours de troupeaux dans les sectionnaux de St Georges de Lévejac ".Rapport de stage à l'INRA ESR, avril 1999, 24p

[5]PROTON, H., BOUSQUET, F., REITZ, P., 1997. un outil pour observer l'organisation d'une société d'agents- le cas d'une société d'agents chasseurs agriculteurs, acte des 5ème Journées francophones JFIADSMA'97 La Colle sur Loup, HERMES Paris Ed., pp159-172.

[6]BOUSQUET F., BAKAM I., PROTON H., LEPAGE C., " Cormas : Common-Pool Resources and Multi-Agent Systems ", Actes de la 11e Conférence Internationale sur les applications industrielles et d'ingénierie de l'Intelligence Artificielle et des Systèmes Experts, Benicàssim, Castellon, Espagne, 1-4 juin 1998, Lecture Notes in Artificial Intelligence, n° 1416, p. 826-837, Springer, Berlin

[7]ROUCHIER, J., BARRETEAU, O., BOUSQUET, F., PROTON, H., 1997. Evolution and co-évolution of individuals and groups in environment, 14 p., ICMAS, International Congress for Multi-agent systems, Paris été 1998

[8]BOUSQUET F., GAUTIER D., " Comparaison de deux approches de modélisation des dynamiques spatiales par simulation multi-agents : les approches spatiales et acteurs ", CyberGéo, http://www.cybergeo.presse.fr/modelis/bousquet/bousquet.htm

[9]Bah, A., Canal, R., D'Aquino, P. and Bousquet, F. 1998. Application des systèmes multi-agents et des algorithmes génétiques à l'étude du milieu pastoral sahelien. In: N. Ferrand (Ed). Modèles et systèmes multi-agents pour la gestion de l'environnement et des territoires. Cemagref Editions. Pp: 207-220.